# Visualization of One-Mode Network Data via Package sna

The xUCINET software itself does not include tools for network visualization. At least two other R packages, sna and igraph, do accomplish this. Both have numerous useful features, but it appears that sna is somewhat less complicated to use; among other things, it can operate directly on data in xUCINET data projects without any need to make further transformations. These notes therefore focus on using sna; we may add some guidance about igraph at a later time.

One begins as usual, by activating the package(s) one needs. Here, this means both xUCINET and sna; install the latter `(install.packages("sna"))` if you didn't do so before while setting up R and xUCINET.

```
library(xUCINET)
library(sna)
```

## Obtaining a very basic visualization

You can visualize a network very quickly, using all default settings (including, importantly, the Fruchterman-Reingold layout algorithm), via the following command:

```
gplot(<network name>)
```

For the Bank Wiring Room "games" network, this would be

```
gplot(Hawthorne_BankWiring$Games)
```

This display can be elaborated in numerous ways. We will introduce several of these in stages.

## Some useful housekeeping

If you try out the above, you will get a picture, but it has some shortcomings, including (1) it is rather small; (2) although it is for an undirected network, it includes arrows at the end of all lines, which add distracting clutter; and (3) it lacks anything to identify the nodes/vertices. These limitations may be overcome by adding a few more lines to the gplot command:

```
par(mar = c(0, 0, 0, 0))
set.seed(02138)
gplot(Hawthorne_BankWiring$Games,
gmode = "graph",
displaylabels = TRUE
)
```

Remarks on added material:

```
par(mar = c(0, 0, 0, 0))
```

An R graphics command that sets the margins of the display window. Setting these to 0 allocates as much space as possible to the picture. The four numbers refer, respectively, to the margins at the bottom, left, top, and right of the display window

```
set.seed(02138)
```

This is an R command that sets the "seed" of the random number generator used with the default (Fruchterman-Reingold) layout algorithm used by gplot. This command is not strictly necessary, but without it you will get a somewhat different picture each time you execute the gplot command. We recommend it while you are developing a visualization. You can put any number you like in the parentheses; I have used the ZIP code for the Harvard FAS campus.

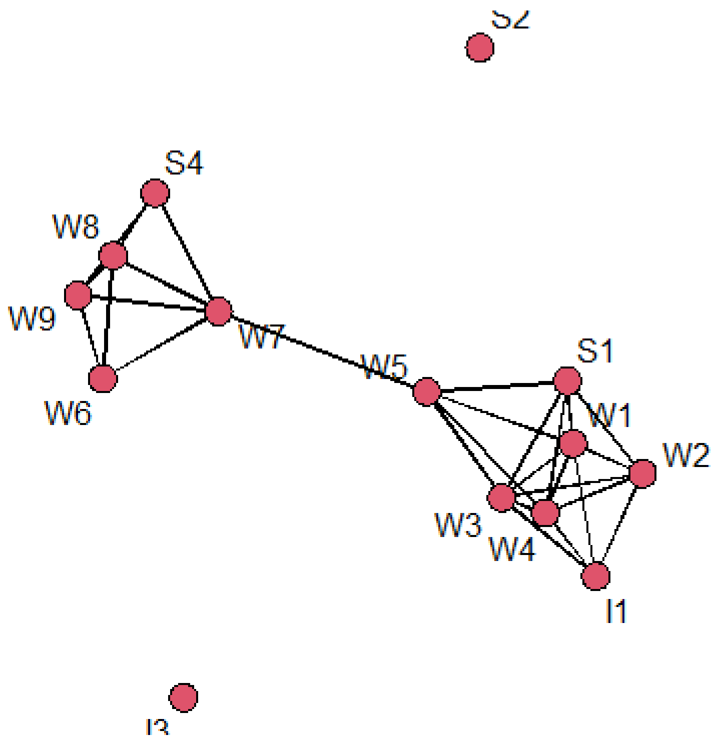
```
gmode = "graph",
```

A gplot argument, the default setting of "gmode" is "digraph", indicating that a network is directed; with it, arrows are displayed. Setting gmode to "graph" is appropriate for an undirected network, and suppresses the arrows

```
displaylabels = TRUE)
```

Another gplot argument, the default setting of "display labels" is "FALSE" meaning that labels are not shown; declaring this to be "TRUE" displays labels for the nodes

With these modifications, the following graph results:



This improves on the default layout and is mostly satisfactory, although the labels for the isolate vertices S2 and I3 are cut off a bit; this problem could be rectified by altering the positioning of the labels, or by increasing the margin settings in the "`par`" command to small fractions.

Controlling the appearance of nodes

To alter the color, shape or size of the nodes/vertices in a display, you can manipulate other gplot arguments. Continuing the example, we add three arguments, one for each of these features:

```
par(mar = c(0, 0, 0, 0))
set.seed(02138)
gplot(Hawthorne_BankWiring$Games,
gmode = "graph",
displaylabels = TRUE,
vertex.cex = 2,
vertex.col = “blue”,
vertex.sides = 4
)
```

Remarks on new material:

```
vertex.cex = 2,
```

This argument controls vertex size, relative to an internally-determined default size. It is an “expansion factor” (hence the “ex” in the argument name). A value greater than 1 increases vertex size, a fractional value decreases it, and a value of 0 yields a blank symbol. Here I’ve asked for symbols twice as large as the default size

```
vertex.col = “blue”,
```

This argument alters the color of vertices. You can insert your favorite color here. You can also give a number from an R color palette (a default palette will be used unless you elect to alter it). The default color (red) is color 2 on the default palette

```
vertex.sides = 4
```

This argument controls vertex shape. The vertices are always polygons. You obtain a shape that is circular (to the eye) by using a large entry like 50 here. Here I’ve asked for diamond shapes.

## Label and Border Appearance

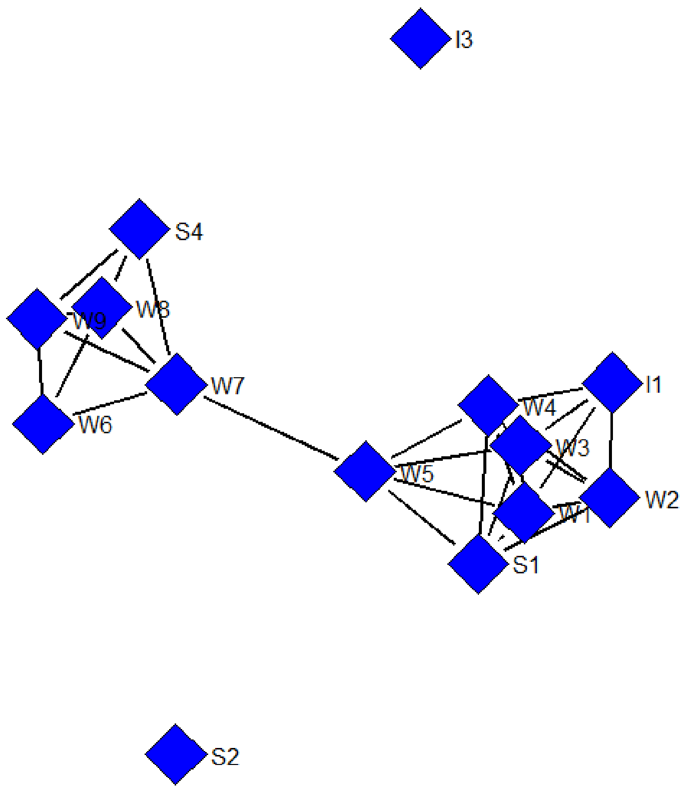
After making these changes, the problem with the labels for the isolate nodes gets worse. Adding an additional argument can fix this:

```
label.pos=4
```

The default for this is 0, which positions labels “optimally” and certainly away from the center of the vertex symbol; other options are 1 (below), 2 (left), 3 (above), 4 (right), 5 (over the vertex). I’ve placed labels to the right of each node by selecting 4.

There are also `label.cex` and `label.col` arguments that govern label size and color; they operate in a manner like `vertex.cex` for vertex size and `vertex.col` for vertex color. Among other gplot options are series of arguments that can customize the appearance of the vertex borders (those shown in black here). I don’t make much use of these, since the borders are small and thin enough that it doesn’t seem wise to use these to communicate information about the network.

With these changes, the redrawn BWR games network looks like this:



When changes to the size, color, or shape of nodes are applied globally, as here, their use is really a matter of aesthetics. They can be valuable, though, in communicating information about differences between the nodes based on attribute data.

Illustrating Attribute-Related Differences between Nodes

Instead of setting `vertex.cex`, `vertex.col` and `vertex.sides` at constant values, one may instead assign vectors of attribute values to these arguments, so that differences in the size, color, or shape of nodes reflect between-node heterogeneity.

The Bank Wiring Room data project in xUCINET doesn't include any attributes (although *Management and the Worker* may describe some), but I've devised one based on the three types of jobs that the employees were performing: as inspectors, wiremen, and soldermen. (This is for the illustrative purposes of these notes only; the node labels already convey this information.) Rather than adding job to the data project as an attribute, I just create a vector in R:

```
job<-c(rep(1,2),rep(2,9),rep(3,3))
```

which yields a vector that looks like this:

```
> job
 [1] 1 1 2 2 2 2 2 2 2 2 2 3 3 3
```

Value 1 is for inspectors, 2 for wiremen, and 3 for soldermen. (Note that a vector like this has to reflect the order in which nodes appear in the data project.)

Then we can use this vector to let the size, color, and shape of the nodes differ by job, via the following gplot command:

```
par(mar = c(0, 0, 0, 0))
set.seed(02138)
gplot(Hawthorne_BankWiring$Games,
      gmode = "graph",
      displaylabels = TRUE,
      vertex.cex = 0.5*job,
      vertex.col = job,
      vertex.sides = 2+job,
      label.pos=4
)
```

Remarks on the vertex-related arguments:

```
vertex.cex = 0.5*job,
```

Assigns expansion factors of 0.5, 1, and 1.5 to inspectors, wiremen, and soldermen, respectively, so soldermen nodes should be largest and inspector nodes smallest

```
vertex.col = job,
```

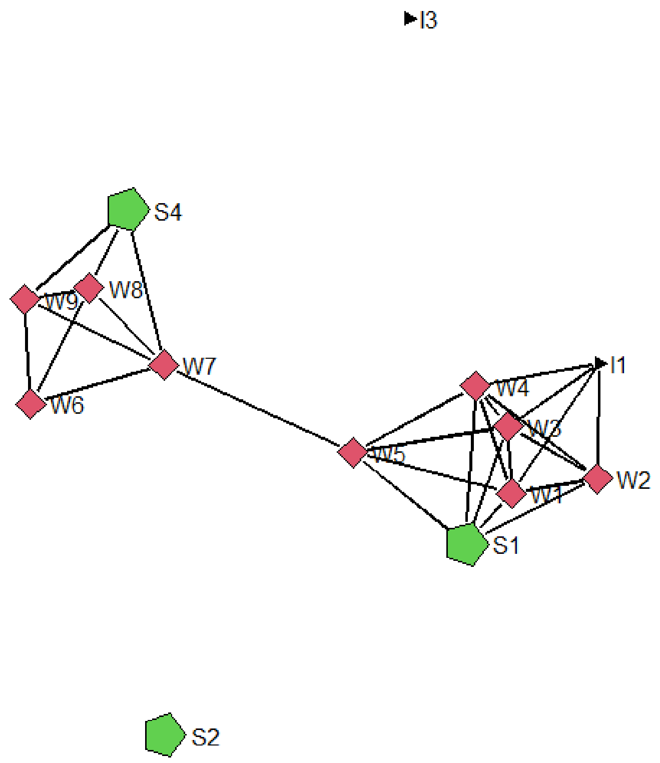
Assigns default color palette colors 1 (black), 2 (red) and 3 (green) to inspectors, wiremen, and soldermen, respectively

```
vertex.sides = 2+job,
```

Assigns shapes of 3 (triangle), 4 (diamond) and 5 (pentagon) to inspectors, wiremen, and soldermen, respectively

Note that it is important to be mindful of the numerical values of attributes when working with vectors like this. (Had these attribute data been added as to the `Hawthorne_BankWiring` data project as an attribute, one would reference them as `Hawthorne_BankWiring$Attribute$job` in the arguments shown above.)

Running this command leads to this refinement of the graph:



The way in which the sizes, colors, and shapes of vertices have changed is evident and corresponds to what was described above.

Ordinarily, of course, there would be no need to make all of these changes to illustrate the same job-related variation among workers; the picture now does that in four ways, through vertex sizes, colors, shapes, and labels. One of these will usually suffice for any given attribute. But when working with other data sets that include more attributes, one can use different attributes with these arguments and hence illustrate several kinds of between-node heterogeneity simultaneously.